CRUD with MongoDB

Starting the database server

Start database with

```
> mongod
```

Log in to mongo

```
> mongo
```

Managing databases

A list of all databases can be retrieved by issuing

```
> show databases
```

If you want to create a new database you simply use the database. However, note that the database is not created before you actually insert data into it.

```
> use todo; // semicolon is optional but good style
```

To check which database you are in you can simply type

```
> db // ‘todo’
```

or

```
> db.getName();
```

To drop a previously created database use it and perform

```
> db.dropDatabase();
```

Collections

Collections are similar to RDBMS’ tables although they are nested. To create a collection, again you simply insert data into it (it’s schema-free after all).

CRUD

To insert data into the collection “tasks”:

```
> db.tasks.insert({ title: 'Test', estimate: 2})
```

To see which collections are in the currently selected database type

```
> db.getCollectionNames();
```

The SQL “SELECT” translates into MongoDB’s “find”:

```
> db.tasks.find({query});
```

If you only one to get a single item you can also use findOne.

```
> db.tasks.findOne({query})
```

Removing is simple too. Furthermore you can specify a limit to delete only a single entry when found.

```
> db.tasks.remove({query}, {justOne: true/false})
```

Updating:

```
> db.tasks.update({query}, {update}, {options});
```

Queries

Queries are used when selecting a document. It basically is the NoSQL version of the WHERE-clause.

An operator is always nested into the field name. e.g. selecting documents where “estimate” is greater than 2:

```
{ estimate: { $gt: 2 } }
```

You can also use JavaScript to query:

```
{ $where:
    function() { return (this.estimate > 5); }
}
```

To combine operators:

```
{ $and: [
    {estimate: { $gt: 3} },
    {estimate: { $lt: 6} }
    ]
}
```

Updates

An update on a document can either consist of operators or be a full update (i.e. replacement). For a replacement just specify the new object. For an update you can use the following operators and more:

$inc

Increase value

```
{$inc: { count: 1 }}
```

$rename

Renames a field

```
{$rename: { 'name': 'fullname' }}
```

$set

Sets a field. In fact, replacement uses this operator.

```
{$set: { name: 'Daniel', age: '19' }}
```

$unset

Unsets a field

```
{$unset: { nickname: '' }}
```

$push

Add an item to an array

```
{$push: { tags: 'code'}}
```

However you can only add one field at a time.

```
{$push: { tags: ['code', 'design'] }}
// results in
// ['blog', ['code', ‘design']]
```

To add more items use the $each modifier

```
{$push: { tags: { $each: ['code', 'design'] }}}
```

$pop

Removes an item from the front/back of the array

```
{$pop: {scores: -1}} // first element removed
```